

APL/S: An Alternative

Robert G Brown, 777 S Mathilda Ave Apt D148,
Sunnyvale CA 94087

About the Author

Robert G Brown is an independent consultant after having worked for IBM for 13 years. He began using APL in 1968 and structured programming in 1970. When he designed the APL/S language in 1978, he attempted to combine structured programming with APL on a small computer in a way which removes some of the common objections to both.

APL/S is a modified subset of APL plus structured-programming control figures. It is intended to be a good first language both for those who may go on to more powerful languages such as Pascal or APL, and for those whose computational needs are destined to remain modest.

Pseudocode

Structured programming is a collection of techniques that help produce demonstrably correct programs. One of the fundamental ideas is to first state the action of the program in what is sometimes called *pseudocode*, or structured English, then progressively refine the statements of the program toward the programming language being used.

With structured English, any imperative statement can be used, but alternation and repetition statements are restricted to a few well defined forms. In this case, the forms are IF-THEN-ELSE, ENDIF for alternation and WHILE-DO, ENDWHILE for repetition. Their intuitive meaning can be illustrated by a set of dieting instructions.

The instructions are to keep eating, one byte at a time, as long as you are hungry. When you are no longer hungry, ask yourself whether you want to get fat. If you do, eat some more. A structured English statement of these instructions is:

```
WHILE you are hungry
DO eat a byte
ENDWHILE
IF you want to get fat
THEN eat some more
ENDIF
```

An APL/S Program

The pseudocode for a guess-the-number-game program is shown in listing 1a. The input is a series of guesses at a

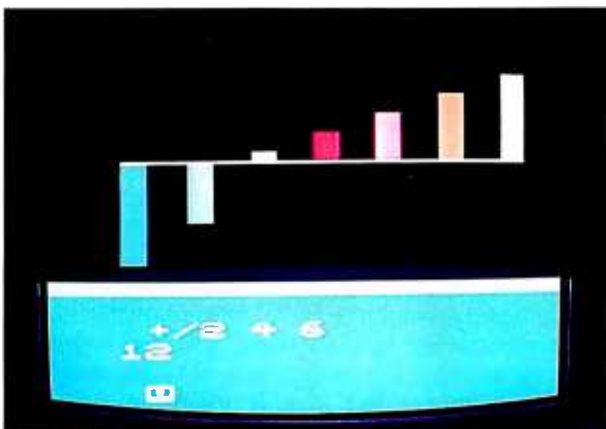


Photo 1: An immediate-mode calculation displayed on an ordinary television set. The user entered "+/2 4 6". This expression was evaluated, and the result (12) was displayed. The +/ characters indicate that the elements of the array 2 4 6 should be added. Therefore 2+4+6 gives 12 as the result. The cursor (an inverse video U, which shows up as a white square with a U inside) indicates that the keyboard is open for the next immediate-mode entry. The histogram bars are left over from a previous calculation.



Photo 2: Program AVE, which computes the average of a set of numeric observations. Input is from the keyboard as a series of numeric values — an array constant. The input, returned by the KEYB function, is assigned to OBS, thus making OBS an array. The average value is computed by adding up the elements of the array (+/OBS) and dividing the sum by the size of the array (SIZE OBS). Since the result is not assigned to a variable, it is displayed.

Listing 1a: Pseudocode for a guess-the-number-game program. This type of expression may be called *Structured English*. See listing 1b for the APL/S equivalent.

```
Pick a number between 1 and 100
Set the number of guesses to 1
WHILE the guess from the keyboard is not equal
  to the number picked
DO add 1 to the number of guesses
  IF the guess is higher than the number
  THEN display "Too high"
  ELSE display "Too low"
  ENDIF
  Display "Try again"
ENDWHILE
Display "Number guesses-"
Display the number of guesses
```

randomly selected number. The outputs are messages saying "too high," "too low," and "try again." When the number is correctly guessed, the number of guesses made is displayed.

The corresponding APL/S program, named GUESS, is shown in listing 1b. It is a line-for-line translation of the pseudocode. By looking at the pseudocode and the sample execution in listing 2, the programmer should be able to understand the main points of the APL/S program, although all the details may not be clear.

A Description of APL/S

A more complete and precise description of APL/S is provided by the syntax diagrams in figure 1 thru 5 and

Listing 1b: APL/S program for the guess-the-number game. In this implementation of the language, keywords such as PROGRAM and RANDOM may be abbreviated by the first four letters. See listing 2 for an example of the execution of this routine.

```
PROGRAM GUESS
  NUM=RANDOM 100
  NGES=1
  WHILE NUM NE GES=KEYB
    DO NGES=NGES+1
      IF GES GT NUM
        THEN "TOO HIGH"
      ELSE "TOO LOW"
      ENDIF
      "TRY AGAIN"
    ENDWHILE
    "NUMBER GUESSES-"
  NGES
ENDPROGRAM
```

Listing 2: Example execution of the program of listing 1b. Execution of a program is started by typing the name of the program, in this case GUESS. User input is prompted by a question mark (?) character.

```
GUESS
?50
TOO HIGH
TRY AGAIN
?25
TOO LOW
TRY AGAIN
?32
NUMBER GUESSES-
3
```

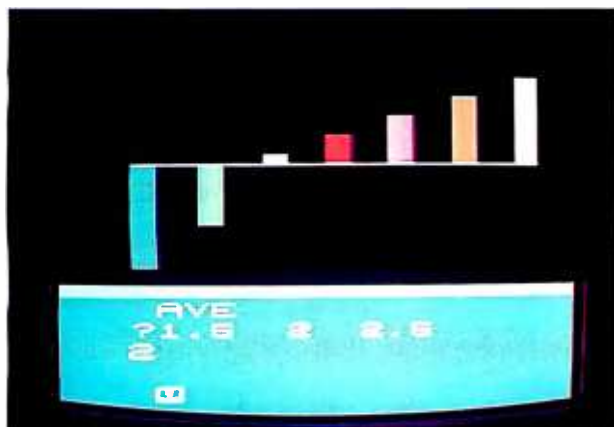


Photo 3: The program AVE is executed by keying in its name. Keyboard input is prompted by a question mark (?). The user entered the 3-element array constant 1.5 2 2.5. In the program, $\div/\text{OBS} \div \text{SIZE OBS}$ was evaluated as $\div/\text{OBS}$ (giving 6) divided by SIZE OBS (giving 3) for a result of 2.

the function descriptions in table 1. Starting at the diagram labeled Program, a path through the diagrams defines a syntactically correct APL/S program. The diagrams do not specify APL/S action (that is, the meaning or semantics of the program).

Each circle indicates a *terminal symbol*. Terminal symbols become part of the program text exactly as shown. The boxes indicate *nonterminals*. For each nonterminal there is a diagram (or, for functions, a description in table 1). When all nonterminals have been replaced with sequences of terminals according to the diagrams (or table 1), the result is a syntactically correct APL/S program.

The syntax diagrams in figures 1 thru 5 do not cover statements for loading, saving, editing, or tracing the execution of programs. A complete description of APL/S can be found in the paper "An Introduction to APL/S" (part of the *Conference Proceedings of the Third West Coast Computer Faire*, November 1978) and the *APL/S User's Manual* by W Judd and S Cintz (available from the VideoBrain Computer Co).

HOLIDAY SPECIAL

SAN DIEGO COUNTY'S

COMPUTER STORE INTL.

Presents the
COMPLETE HOME COMPUTER SYSTEM



REG. \$2,350

NOW \$1,895

- Integer or Apple Soft Basic
- 32K User Memory (RAM)
- Color Graphics
- Music and Sound Effects
- Excellent Reliability

APPLE DISK II SYSTEM

- Disk Drive with Controller
- Disk Operating System
- 110 K On Line Storage
- 10 Pre-Programmed Diskettes

13" PORTABLE T.V. SET

TO TAKE ADVANTAGE OF COLOR GRAPHICS, WITH RF TV MODULATOR



COMPUTER STORE INTL.
A DIVISION OF COMPUTER METRICS INC.

**1251 BROADWAY
EL CAJON CA. 92021
(714) 579-8066**

OFFER EXPIRES 12-31-79

	APL/S	APL	Definition
Scalar 2-argument functions	$X + Y$ $X - Y$ $X \times Y$ $X \div Y$ $X \uparrow Y$ $X \text{ MAX } Y$ $X \text{ MIN } Y$ $X \text{ MOD } Y$ $X \text{ LOG } Y$ $X \text{ LT } Y$ $X \text{ LE } Y$ $X \text{ EQ } Y$ $X \text{ GE } Y$ $X \text{ GT } Y$ $X \text{ NE } Y$ $X \text{ AND } Y$ $X \text{ OR } Y$	$X + Y$ $X - Y$ $X \times Y$ $X \div Y$ $X \uparrow Y$ $X \Gamma Y$ $X L Y$ $X Y$ $X \odot Y$ $X < Y$ $X \leq Y$ $X = Y$ $X \geq Y$ $X > Y$ $X \neq Y$ $X \wedge Y$ $X \vee Y$	X plus Y X minus Y X times Y X divided by Y X to the Yth power maximum of X and Y minimum of X and Y X modulo Y X residue of Y base X log of Y X less than Y X less than or equal to Y X equal to Y X greater than or equal to Y X greater than Y X not equal to Y X and Y X or Y
Scalar 1-argument functions	$+Y$ $-Y$ $\text{SIGN } Y$ $+Y$ $\text{EXP } Y$ $\text{CEIL } Y$ $\text{FLOOR } Y$ $\text{ABS } Y$ $\text{LOG } Y$ $!Y$ $\text{RAND } Y$ $\text{NOT } Y$	$+Y$ $-Y$ $\times Y$ $+Y$ $\star Y$ ΓY $L Y$ $ Y$ $\odot Y$ $!Y$ $?Y$ $\sim Y$	Y negative of Y signum of Y reciprocal of Y e to the Yth power ceiling of Y floor of Y absolute value of Y natural log of Y factorial of Y a random integer from 1 to Y not Y
Mixed 2-argument functions	X, Y	X, Y	the concatenation of X and Y, an array
Mixed 1-argument functions	$\text{SIZE } Y$ $\text{INDEX } Y$ $\text{ARRAY } Y$ $.Y$ $X(Y)$ $X = Y$ $\odot / Y$	ρY ιY $Y \rho 0$ $.Y$ $X[Y]$ $X - Y$ $\odot / Y$	number of elements in Y 1,2,...,Y size Y array of zeros Y ensured a vector the element of X selected by Y X assumes the value of Y the $\odot$ reduction of Y, where $\odot$ is any scalar 2-argument function: $Y_1 \odot Y_2 \odot \dots \odot Y_n$
Subscripting	$X(Y)$	$X[Y]$	the element of X selected by Y
Assignment	$X = Y$	$X - Y$	X assumes the value of Y
Reduction operator	$\odot / Y$	$\odot / Y$	the $\odot$ reduction of Y, where $\odot$ is any scalar 2-argument function: $Y_1 \odot Y_2 \odot \dots \odot Y_n$
Input	KEYB	$\square$	keyboard input
Output	X "X"	X 'X'	display the value of X display X
Circle functions (scalar 1-argument functions in APL/S)	$\text{SIN } X$ $\text{COS } X$ $\text{TAN } X$	$1oX$ $2oX$ $3oX$	$\sin X$ $\cos X$ $\tan X$

Table 1: APL/S functions with equivalent APL functions shown for comparison.

Because of the limited character set available, the relational functions are denoted with alphabetic symbols (eg: NE for $\neq$) and assignment is denoted by $=$ instead of the preferable $\leftarrow$ of APL. In most other cases, functions which are denoted in APL by special characters are denoted in APL/S by the name of the function as given in descriptions of APL.

In APL/S, mathematical formulas are evaluated *left to right* with addition and subtraction done last. For example, $2*4 + 10 \times 3 - 1$ is evaluated as follows:

$$\begin{aligned}
 &2*4 + 10 \times 3 - 1 \\
 &16 + 10 \times 3 - 1 \\
 &16 + 30 - 1 \\
 &46 - 1 \\
 &45
 \end{aligned}$$



Figure 1: Syntax diagram showing global structure of an APL/S program.

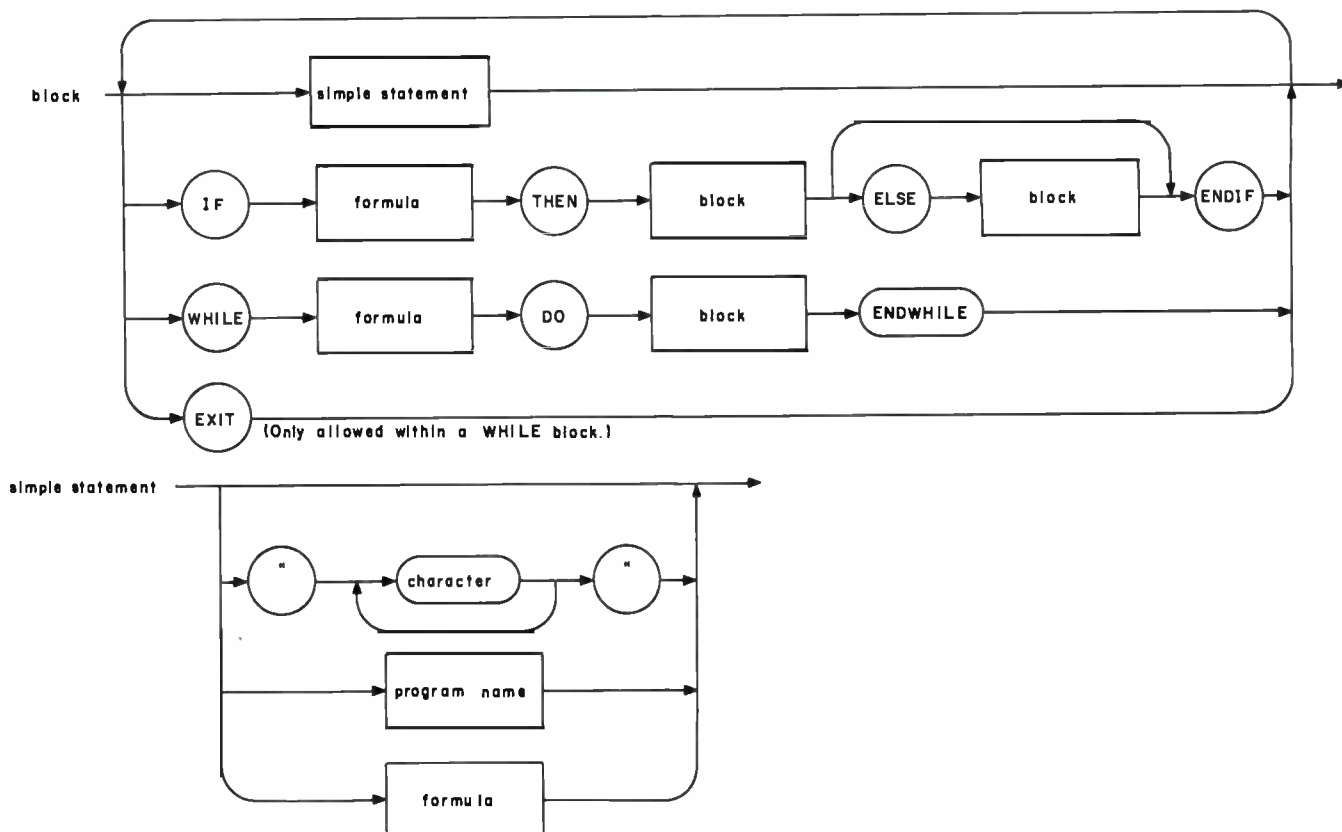


Figure 2: Syntax diagram for block and simple statement structures.

For an assignment statement, evaluation cannot be strictly left to right. For example, the statement $A = 10 + 1$ is evaluated by first adding 10 and 1 for a result of 11, then assigning the 11 to A.

Evaluating the right hand side of an assignment before performing the assignment is carried forward to the case of an assignment embedded within a formula. For example:

```

3 × A = 10 + 1
3 × A = 11
3 × 11
33

```

The use of assignment within a formula is illustrated in the program GUESS of listing 1b in the line `WHILE NUM NE GES=KEYB`. The evaluation of the line goes as follows: the KEYB function reads the keyboard and returns the value entered, the assignment operation places the value into the variable GES, and the NE function is then evaluated to 1 if NUM is not equal to GES or to 0 if NUM is equal to GES. WHILE will then have an argument of 0 or 1.

WHILE, IF, AND, OR, and all the relational functions treat 1 as true and 0 as false. APL/S makes no data type distinctions such as boolean, integer, or real — a number is a number. Functions such as AND are defined on a subset of the numbers, namely 0 and 1.

The control figures of APL/S are Sequence; IF-THEN-ELSE, ENDIF; WHILE-DO, ENDWHILE; and Sub-

programs. There is no GOTO statement, but there is an EXIT for early termination of a WHILE loop. Because all control figures are self terminating (ie: ENDIF, ENDWHILE, ENDPGRAM), there is no need for BEGIN-END pairs to form compound statements. Anywhere a single statement can appear, so can a block of statements.

Programs can be invoked recursively. The major weakness is that all variables are global. These facilities of the language are powerful enough that the popular eight-Queens problem can be solved by an APL/S program which closely follows the recursive and well structured solution given by Dijkstra. (The differences are array bounds and local variables.)

The eight-Queens problem was discussed from a beginner's viewpoint in an article in the October 1978 BYTE ("Solving the Eight-Queens Problem," by Terry Smith, page 122) and from a more sophisticated vantage by several readers in the February 1979 BYTE ("Eight-Queens Forum," pages 132 through 148). Dr E W Dijkstra's solution is found on pages 72 thru 82 of *Structured Programming*, by Dahl, Dijkstra, and Hoare (Academic Press, 1972):

Like APL, in APL/S all scalar functions are extended element-by-element to arrays, any scalar two-argument function can be used to reduce an array, and mixed functions such as SIZE (ρ in APL) are defined. Unlike APL, in APL/S arrays are restricted to one dimension, and subscript expressions must evaluate to scalars (or one-element arrays). Some examples should help clarify the array features.

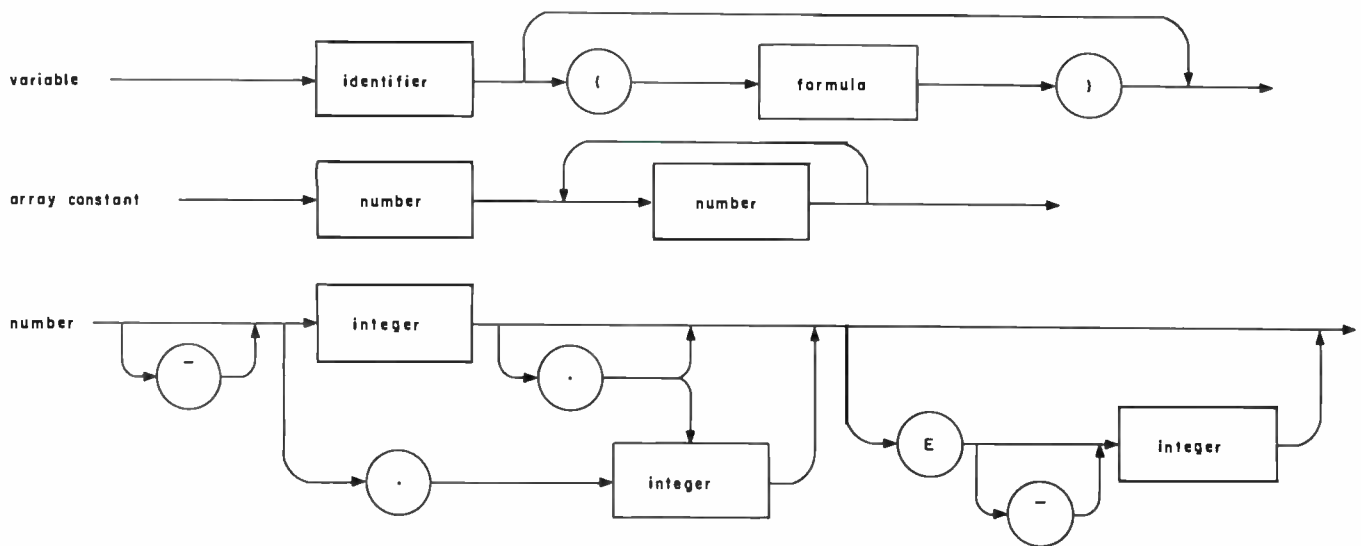


Figure 3: Syntax diagram for variable, array constant, and number structures.

Some Examples

Any *simple statement* (as defined in figure 2) can be entered for immediate evaluation. Photo 1 shows an example of an immediate mode calculation on the VideoBrain computer. The user entered `+ /2 4 6`. It was evaluated and the result, 12, was displayed. The "2 4 6" is

an example of an array constant. Addition was used to reduce the array to a scalar. Reduction by addition (`+/`) can be visualized as: `+ /2 4 6` gives `2+4+6` gives 12.

A more realistic use of arrays in immediate mode is the calculation of *net present value*. Given the one-dimensional array of cash flows, $C = (C_1, C_2, \dots, C_n)$, the net present value (NPV) at an interest rate I , is given by:

$$NPV = \sum_{j=1}^{j=n} \frac{C_j}{(1+I)^j}$$

In APL/S, the formula for the net present value is:

$$+/(C \div ((1+I) * \text{INDEX SIZE } C))$$

For comparison, the equivalent formula in APL is:

$$+ / C \div (1+I) * \iota \rho C$$

The two formulas are similar, differing only in function names and parentheses (to insure the same order of evaluation).

Because this kind of use of the array facilities is at the very heart of APL or APL/S programming, it is essential to understand how such formulas are evaluated. Let $I=0.1$ and $C = ^100\ 50\ 150$. The evaluation of the APL/S formula can be traced through its intermediate results:

$$\begin{aligned} &+/(C \div ((1+I) * \text{INDEX SIZE } C)) \\ &+/(^100\ 50\ 150 \div (1.1 * \text{INDEX SIZE } ^100\ 50\ 150)) \\ &+/(^100\ 50\ 150 \div (1.1 * \text{INDEX SIZE } 3)) \\ &+/(^100\ 50\ 150 \div (1.1 * 1\ 2\ 3)) \\ &+/(^100\ 50\ 150 \div (1.1\ 1.21\ 1.331)) \\ &+/(^90.909\ 41.32\ 112.7) \\ &^90.909 + 41.32 + 112.7 \\ &63.111 \end{aligned}$$

Text continued on page 98

SPECIAL INTRODUCTORY OFFER



ANADEx DP-8000
LINE PRINTER

LOGON offers you this excellent small reliable printer at a low introductory price. Every DP-8000 is complete with the latest features: - Adjustable tractors and 1K input buffer. Standard features include:

- RS232C, current loop and centronics parallel
- 80 columns - 112 char/sec
- 96 char set - 9 x 7 font — Bidirectional printing
- Top of form, skip over perf, out of paper, eight vertical tabs, etc.

Only **\$795**

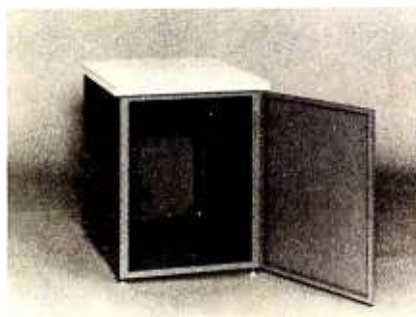
Plus \$20.00 shipping and handling.
Terms - check with order - Allow 3 weeks for delivery



Logon Inc.
246 O Lemoine Ave.
Fort Lee, N.J. 07024
201-224-6911
212-594-8202

Circle 58 on inquiry card.

DESKS AND STUFF



Computer terminals, business systems, lab components . . . they all need desks and enclosures. That's what we're all about. Computer Furniture and Accessories offers a standard line of furniture suitable for a wide variety of applications. Handsome, rugged, economical furniture in all shapes, sizes and colors. Basic models shipped from stock in days, not months. And we're nice people to deal with. What more could you ask for?

CF&A

**Computer Furniture and
Accessories, Inc.**
1441 West 132nd Street
Gardena, CA 90249
(213) 327-7710

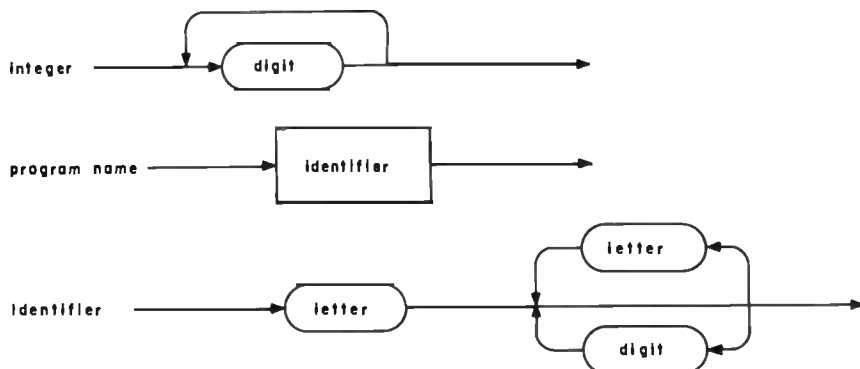


Figure 4: Syntax diagram showing integer, program name, and identifier structures.

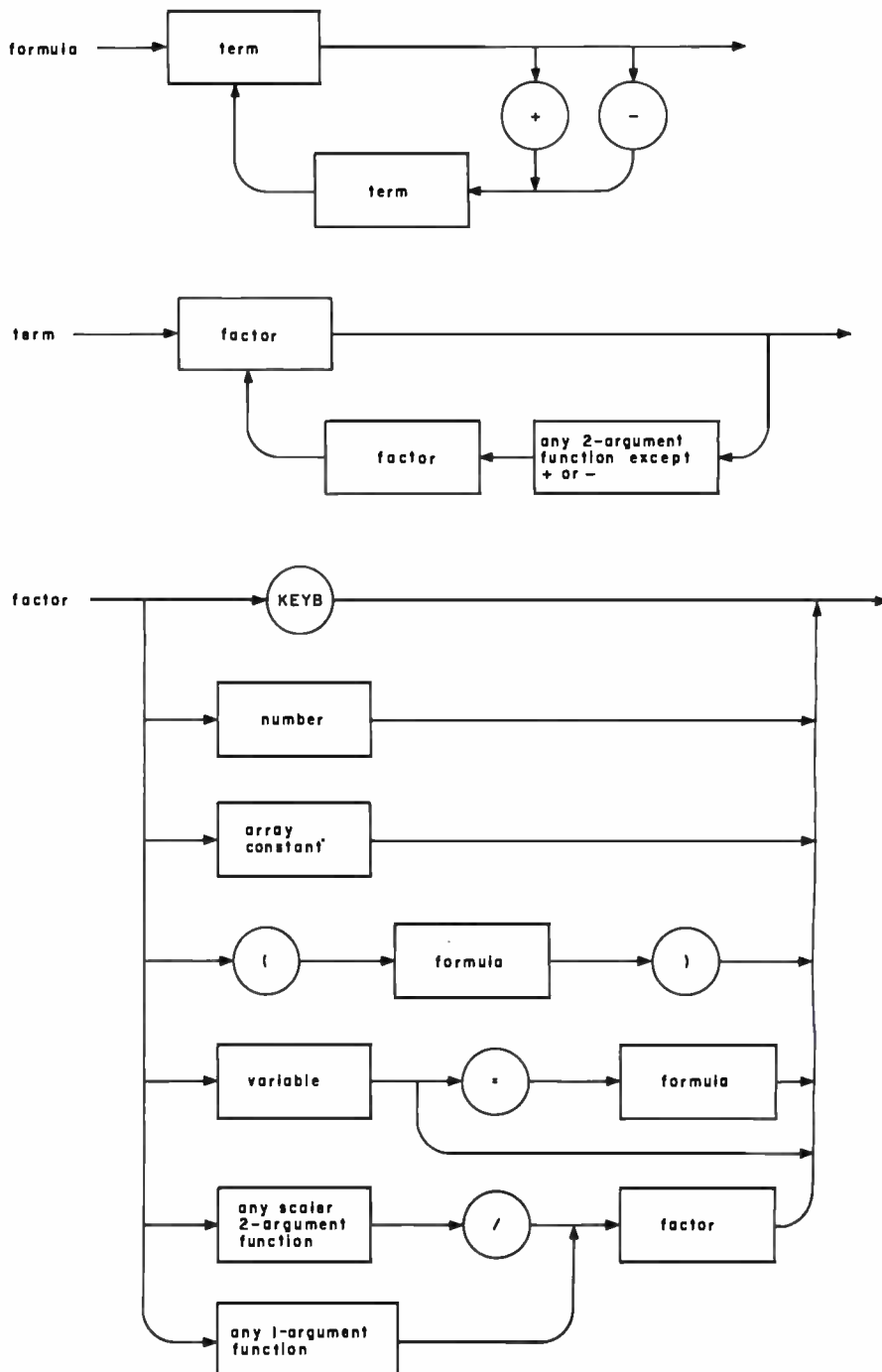


Figure 5: Syntax diagram of formula, term, and factor structures.

Listing 3: An APL/S program that simulates a game of craps. The game continues until a bet of \$0 is made. Actual play is handled by the subroutine PLAY, which sets WIN to 1 to indicate a win, and to 0 for a loss. The net amount won is stored in NET.

```
PROGRAM CRAPS
  NET=0
  "BET 0 TO QUIT"
  WHILE 1
    DO "PLACE YOUR BET"
      IF 0 GE BET=KEYB
      THEN EXIT
      ELSE "COMING OUT"
        PLAY
        IF WIN
        THEN "WINNER"
          NET=NET+BET
        ELSE "YOU LOSE"
          NET=NET-BET
        ENDIF
      ENDIF
    ENDWHILE
    IF NET GE 0
    THEN "YOU'VE WON"
      NET
    ELSE "YOU'VE LOST"
      -NET
    ENDIF
    "COME.AGAIN.SOON"
  ENDPROGRAM
```

Listing 4: The PLAY subroutine used by the CRAPS program of listing 3. A first roll of 7 or 11 wins; 2, 3, or 12 loses. Otherwise, a point is established (the value of the first roll). The dice are then repeatedly rolled until the point is rolled for a win, or a 7 is rolled for a loss. See listing 5 for a sample execution of CRAPS.

```
PROGRAM PLAY
  +DICE=RANDOM 6 6
  PONT=+/DICE
  IF OR/(PONT EQ 7 11)
  THEN "PASS"
    WIN=1
  ELSE WIN=0
    IF OR/(PONT EQ 2 3 12)
    THEN "CRAPS"
      ELSE "YOUR POINT IS"
        PONT
        WHILE 1
          DO +DICE=RANDOM 6 6
            IF +/DICE EQ PONT
            THEN "POINT MADE"
              WIN=1
              EXIT
            ELSE
              IF +/DICE EQ 7
              THEN "BUSTED"
                EXIT
              ENDIF
            ENDIF
          ENDWHILE
        ENDWHILE
      ENDIF
    ENDIF
  ENDPROGRAM
```

Listing 5: Sample execution of CRAPS.

```
CRAPS
BET 0 TO QUIT
PLACE YOUR BET
?250
COMING OUT
4 3
PASS
WINNER
PLACE YOUR BET
?300
COMING OUT
1 1
CRAPS
YOU LOSE
PLACE YOUR BET
?200
COMING OUT
2 4
YOUR POINT IS
6
1 4
6 2
5 5
1 5
POINT MADE
WINNER
PLACE YOUR BET
?0
YOU'VE WON
150
COME AGAIN SOON
```

Text continued from page 94:

A simple program (AVE) using the array facilities is shown in photo 2 as it would appear to the user. The program computes the average of a set of observations. An execution of AVE is shown in photo 3.

The final example uses both the structured programming and array features for a simple game of craps. The main program is shown in listing 3, a subprogram in listing 4, and a sample execution in listing 5.

Both the main program and the subprogram use an EXIT statement to end a loop based on a condition detected inside the loop. The single entry, single exit convention of structured programming is maintained by this highly restricted type of GOTO.

The subprogram PLAY (listing 4) contains examples of using the array features for logical testing, a use which is not obvious in the numeric computation context usually employed in explaining the array features. Such non-mathematical use of the array features is common in APL programming, although in APL it is used in conjunction with a kind of conditional GOTO. The evaluation of the expression IF OR/(PONT EQ 7 11) will be followed through its intermediate results.

Although the purpose of the statement is logical testing, the execution trace will show how the statement is, in its use of the array features, similar to the expression for net present value. For tracing, let PONT=9.

IF OR/(PONT EQ 7 11)
 IF OR/(9 EQ 7 11)
 IF OR/(0 0)
 IF 0 OR 0
 IF 0

The result, 0, indicates that it is false that PONT equals 7 or 11, so execution will proceed to the first statement of the ELSE block.

Summary

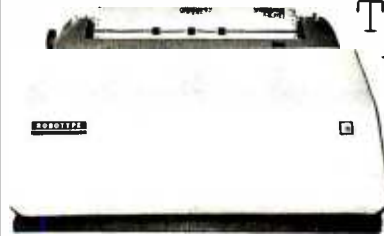
APL/S is one of the first high-level language alternatives to BASIC to be offered on a low-priced personal computer (under \$800). APL/S combines structured programming with an APL approach to arrays. Additional differences between APL and APL/S are due to hardware limitations and a desire to make use of the language as natural as possible. ■

An APL/S language system is available in a read-only memory cartridge for the VideoBrain home computer from VideoBrain Computer Co, 2950 Patrick Henry Dr, Santa Clara CA 95050.

REFERENCES

1. Brown, R G, "An Introduction to APL/S," *Conference Proceedings of the Third West Coast Computer Faire*, November 1978.
2. Dahl, Dijkstra, and Hoare, *Structured Programming*, Academic Press, 1972.
3. Judd, W and Cintz S, *APL/S User's Manual*, Video Brain Computer Co, Santa Clara CA, 1978.
4. Smith, Terry, "Solving the Eight-Queens Problem," October 1978 BYTE volume 3, number 10, page 122.

New Robotype™



Turns your typewriter into a quality output printer!

- easily connects to any computer
- serial and parallel interface
- all electronics included—ready to use from package
- connects to IBM Selectric II typewriter in just one minute
- adapts to a variety of typewriters—no modifications
- compatible with Radio Shack TRS-80, Apple II, Pet, etc.
- Centronics interface compatible
- available from stock in 30 days

Put a Robotype to work on your typewriter for under \$1,000. Call (614) 436-3163 today!

Dealer, distributors, and word processor OEM inquiries welcome.



Robotype™
 (A trademark of Compu-Matics, Inc.)

Applied
 Computer
 Systems, Inc.



77 East Wilson Bridge Road • Worthington, Ohio 43085

Low Power 32K RAM for Heath® H8 computers

DG-32D 32K RAM FEATURES:

- ✓ Plugs into Heath® H8 Computer
- ✓ Ready to use. Fully assembled, tested & burned in
- ✓ Operates with existing Heath memory
- ✓ Protected Memory Output Buffers in the event of Address error.
- ✓ Utilizes popular 4116 RAM devices
- ✓ Memory Address DIP switch changeable
- ✓ Arranged as 4 Independent 8K Blocks
- ✓ Low Power Consumption: Less than 6 watts, typical
- ✓ Transparent Refresh
- ✓ One year guarantee
- ✓ Compatible with all current H8 peripherals.

Heath® and H8 are registered trademarks of the Heath Corporation, Benton Harbor, Michigan.

D·G ELECTRONIC DEVELOPMENTS CO.

\$479

D·G Electronic Developments Co. brings you a totally compatible, fully assembled and tested 32K RAM for Heath® H8 computers. The DG-32D has less than 6 watts power consumption. This allows you to add a full 32K bytes of Random Access Memory without taxing or replacing your computer's power supply. Engineered to plug-in and run without any user modifications, the DG-32D can be used with or without existing H8 RAM without modification. Protection of the memory output buffers is provided in the event of assigning two blocks to the same address space. The DG-32D is the ideal answer to expansion of the Heath H8 computer . . . Low power consumption, low price, high capacity, total engineering and exacting production methods.

Ordering Information: DG-32D RAM available only from DG Electronic Developments Co., P.O. Box 1124, 1827 South Armstrong, Denison, Texas 75020. Check, money-order, VISA or Master Charge. Phone orders accepted on charge orders. NO COD's. Foreign orders add 30%. Texas residents add 5%. For VISA or Master Charge orders call 214-465-7805. \$479.00 freight prepaid.

32K RAM for HEATH® H8